Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 36 (2014) 535 – 540

Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014-Philadelphia, PA

Evolving Vacation Packages: Genetic Algorithms for Entertainment

Iren Valova*[a], Andrew Embry[a], MacKinley Trudeau[a], Gueorgui Gueorguiev[b]

[a]*Computer and Information Science Department, University of Massachusetts Dartmouth, MA 02747, USA*
Computer Science Department, University of Wiscconsin Oshkosh, WI 54901, USA

Abstract

Excitement about your Disney park visit can be easily overshadowed by long lines, nauseous rides, and long days filled with unsatisfied customers around you. Why not employ genetic algorithms - nature's answer to process scheduling and trip optimizations. We discuss GA implementation with variable length chromosomes dynamically created based on the user's input regarding their park visit preferences (length of stay, waiting time, level of nausea, number of rides, etc). We have also accounted for speed-passes, which generates varied initial population. Aside from the initial user preferences, the fitness function takes into account walking distance between rides, whether a trip is child friendly, and preference of a certain type of ride among others. The final solution represents the optimized schedule of rides for the user as well as a few options in case the user has slight shift in preferences. Our experiments are very successful based on a pool of 50 user evaluations of predicted and actual experience.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: genetic algorithms; optimization; scheduling and planning

1. Introduction

Vacations and amusement parks in today's world are complex attractions with many factors which influence the enjoyment of the trip. There are many types of attractions with different levels of intensity, various line lengths and height requirements. Many parks offer speed passes that let visitors skip lines and allows them to visit more

* Corresponding author. Tel.: 5089998502; fax: 5089999144.
E-mail address: ivalova@umassd.edu

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.035

attractions in less time. Visitors can attempt to plan their trip using online resources, however the number of factors that go into scheduling a trip is too large and complicated to be evaluated at just a glance. It might seem like a good idea to go to the closest attraction first, but long lines and the nausea incurred from that attraction could make a speed-pass purchase useless or even ruin your whole visit.

The premise of this problem consists of choosing which attractions to participate in and in what order. The sub-problem of choosing attractions resembles the unbounded knapsack problem. The unbound knapsack problem is a combinatorial optimization problem[1]. It consists of a sack with a limited capacity bound by a maximum weight and *n* number of unique items that have an associated weight and profit. The goal is to maximize the profit in the sack while ensuring the weight does not exceed its capacity[2]. Any of the *n* items can be placed in the sack multiple times, limited by weight, which is equivalent to someone participating in any attraction multiple times, but being limited by time. Since the unbounded knapsack problem is known to be NP-hard, and it is part of our scenario at hand, it follows that our problem is NP-hard as well[3]. The second sub-problem of ordering the attractions is a scheduling problem. Scheduling can be defined as deciding the start times of a sequence of actions where the order of these actions has consequences on the schedule[8]. The part that affects our problem the most is the consequences associated with each action. Although a combination of rides may contain an optimal solution, it will only matter if ordered properly. In respect to the unbounded knapsack problem, the order at which items are added to the sack would affect the profit and/or weight of them. In our problem every attraction visited has consequences on the following attractions in the form of nausea, diminishing returns and physical distance between the attractions.

When considering all the attributes that contribute to one's experience on vacation, the weight of each decision becomes clear. Each attraction provides a specific amount of utility to a visitor based on various factors such as type of attraction, intensity of attraction and the sequence of attractions visited before it. Visitors must also consider how prone to nausea they are, preference to a specific type of attraction over another, whether they enjoy going to the same attraction multiple times, how long they plan to stay at the park and the distance between rides. When planned properly, these complications can work together to minimize nausea, maximize enjoyment and fully utilize the visitors time. We propose using a genetic algorithm with variable length chromosomes to create an optimal itinerary that will accomplish this.

In traditional implementations of genetic algorithms every chromosome must have the same fixed length[10]. This poses many issues for our problem because the length of the chromosome contributes to the solution. In many cases this issue can be avoided by properly and cleverly representing the problem. We found this was not the case in our scenario. Since the order in which the rides are visited matters and because any attraction can be visited multiple number of times at any given time during the day, the only way to effectively represent the problem is by using variable length chromosomes. These have the advantage of dynamic chromosome lengths providing the ability for the algorithms to find an optimal length[11]. It is necessary for this problem to not be bound by visiting a fixed number of attractions because an optimal solution itinerary may require more or less rides.

2. Method

In order to help a user determine a customized itinerary with an optimized attraction schedule, we first ask the user about their preferences. Hence, our approach is user-centric. These preferences include how likely they will become nauseous from a high thrill ride and how quickly the nausea passes after getting off the ride. We also inquire about how likely the user will ride the same ride more than once. This attribute allows us to adjust the total enjoyment from a given ride with a diminishing returns effect. The user selects whether they have kids with them, as this will affect which rides they can go on. They also state if they plan on purchasing a speed-pass to bypass long wait times for popular rides and how long they want to stay. Using these attributes along with attributes to track how much time they have left on their planned trip, an estimate of how nauseous they currently are feeling, and where they are located in the park will be used to evaluate their total enjoyment of the planned trip.

2.1. Genetic algorithm parameters

Traditionally, a chromosome is a sequence of binary digits, which represents a solution in the problem domain[9]. The bits of the chromosome are referred to as its genes and are mapped to parts of a solution[10]. For our purposes, the chromosome is a array of integers which represents a possible itinerary for the day and each integer is a gene. Each gene is associated with a specific attraction and points to the attributes of it using a unique id number (Fig. 1). Each attraction consists of the following attributes:

- Id - Unique identifier for this attraction.
- Type - Category of attraction : {Roller coaster, water ride, etc.}
- Wait Time - Average wait time for standing on line for the attraction.
- Completion Time - Time needed to participate in and complete the attraction.
- Utility - Estimated about of enjoyment the user gets from the attraction.
- Kid Friendly - Used to determine if the ride allows children.
- Nausea Level - Estimated nausea value used to determine how sick the user will get.

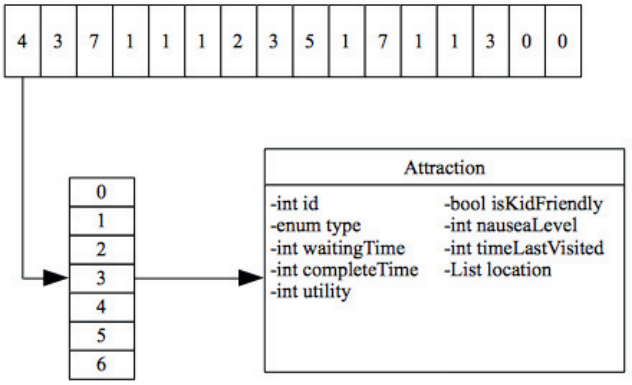

Figure 1 Model of genes in a chromosome referencing an attraction

- Last Visited - Keeps track of the last time the user was at this attraction.
- Location - The area of the park in which the attraction is located. Used to determine the time it takes to travel between two attractions.

There are two special cases when a gene does not represent an attraction. A 0 represents going home for the day and a 1 represents a 5-minute break to help reduce the user's nausea level. For example, the chromosome 2311450 represents traveling to and participating in the attraction with the id of 2 followed by 3, taking a 10 minute break, traveling to and participating in the attraction with the id of 4 followed by 5 and lastly going home.

Chromosome length will vary based on the user's input for how long they want to stay at the park. Each gene represents a minimum of a 5-minute block of time spent at the amusement park, which along with the time the user desires to stay, will determine the actual length of the chromosome used in the algorithm. Where t is the time in minutes the user wants to stay, and x is the minimum of 5 minutes per gene, the number of genes in the chromosome, n, is calculated as follows: $n = t / x$. If the user plans on spending 12 hours at the park, then the chromosome will have a maximum of 144 genes associated with it. A user who specifies only an 8-hour trip will only need a chromosome consisting of a maximum of 96 genes. The number of genes used to evaluate a proposed solution will vary based on when the first 0 gene is located. Since this signifies when the user will go home, only genes to the left of it are used to calculate the total utility for the solution. Since a 0 can appear anywhere in the chromosome, the length that is considered part of the itinerary is allowed to grow and shrink depending on where the first 0 appears, satisfying the condition of variable length chromosomes[11].

The initial population of chromosomes for our genetic algorithm is created randomly with the constraint that the chromosome represents a valid solution. A valid solution would require the chromosome to have a 0 gene representing the user going home. If a chromosome is created without a 0 then it is removed and we try again. The size of the population used for our experiment is 50 chromosomes.

Chromosomes evolve with each generation of the algorithm. We used roulette wheel selection to determine which chromosomes are used with the crossover operator. Roulette wheel selection selects two parent chromosomes at a time to crossover where the fittest chromosomes have a larger chance of being chosen[6]. Single point crossover is used to create two new children chromosomes, from partial solutions of the parents, which are then inserted into the next generation's population. In single point crossover, randomly selected points, called crossover points, between genes are selected on each parent and the genes following those points are swapped between the parents[9]. The only

constraint for the crossover point is that the location of the cut in the chromosome must be before the first 0 gene in the chromosome.

Mutation of genes allows the algorithm to escape local optima through unexpected variation in the population[12]. Each gene in every chromosome has a chance to mutate at every generation by a set mutation rate. In our experiment we used the rate of .05% and swapped the current gene with a new randomly created one if it was selected to mutate. We also implement a dynamic mutation rate to encourage more variation in chromosome length. The chance of mutating the first "go home" gene in a chromosome is increased to promote a longer stay at the park. This allows more unique solutions to be considered in less time therefore increasing the efficiency of the algorithm[7].

When a speed-pass is used to skip long lines at the amusement park, a special case is created within the mutation operator. Since the number of attractions a speed-pass user can visit in a day is greater than a non speed-pass user, a multiplier is used for the "go home" gene to allow the length of the visit to grow quicker than a non speed-pass user. The multiplier used for our experiments was 4, which made the mutation rate of the first go home gene 2.0%.

Sometimes the best chromosome is lost due to mutation and/or crossover, which could reduce the algorithms ability to create better solutions at each generation. To handle this, we implement elite selection of the best possible solution for each generation. With elite selection, the best chromosome in the current generation is automatically carried over to the next generation. This insures the overall fitness of the population will gradually increase[5].

A fitness function is used to evaluate each chromosome during the population evaluation. The function is used to combine different user defined parameters and the various variables to create a fitness value which ranks the chromosomes[13]. The fitness function takes into account the user's inputted nausea tolerance and repeat attraction visitation to determine the total fitness of the chromosome. The evaluation treats the chromosome as a schedule of rides in which the user visits and checks the user's nausea level during the planned trip based on which attractions they are scheduled to visit. The genes of the chromosome are interpreted from left to right and are read one gene at a time. Each gene in the chromosome represents going on the assigned ride. The algorithm needs to keep track of how much utility the user has gained, where the user is located in the park, what time was an attraction last visited, how nauseous the user currently is, and how much time they have spent at the park. To ensure the schedule has the user leave close to their designated time, we keep track of the time spent at the park and implement a penalty function which reduces the total fitness of a solution if it violates the constraint[4]. The penalty function allows for flexibility in the chromosome which permits some schedules that do not fit the time frame exactly but may potentially be an optimal solution. The fitness function adjusts user values at every gene as follows:

1. Time to travel to attraction from current location is added to the time spent: (This is accomplished using Dijkstra's Algorithm; Users location is updated)
2. Attractions wait time is added to the time spent: A check is made prior to see if user choose to get a speed pass and is taken into account by reducing this time to 5 minutes
3. Attractions completion time is added to the time spent
4. After travel time, waiting time and ride time is handled, users current nausea is decreased accordingly
5. The attractions utility is added to users total utility: (Users current nausea is taken into account prior and the utility gained is penalized; Attractions last visited is taken into account prior and the utility gained is penalized depending on how likely they said they are to go on the same ride multiple times; Users time spent at the park is examined prior and if they it exceeds their designated time, the utility gained is penalized)
6. Users current nausea is increased by rides nausea level: Users sensitivity to nausea is accounted for
7. Attractions last visited is updated with current time

When the simulation encounters a 0 it stops and the users total utility is assigned to the fitness of the chromosome.

3. Results

Our experiment used data collected from the New England Six Flags amusement park website which allowed us to apply the algorithm to an actual amusement park. Six flags does not supply information to the public about wait times for rides or how nauseous a ride makes the user, so the authors used personal experience to fill in any missing data with an estimate based upon personal experience. The rides included in this experiment are the water and thrill rides. The kid's rides were excluded from this example for simplicity reasons.

There are a total of 43 rides, which were used to run the algorithm in this experiment. Each ride had a different thrill factor, nausea factor, and wait time associated with it. We set the maximum length of stay to 12 hours to create a realistic day at a park. We said the user had an average tolerance for nauseous rides and also stated the user did not want to ride the same ride twice.

Speed-pass: No; Generations: 5000; Nausea(1-10): 5 - Average nausea tolerance
Ride Diminishing Returns(1-10):1 - Dislikes duplication of rides in the schedule
Fitness of Best Chromosome: 746.4254964375
Decoded Chromosome to Actual Rides:

1 Catwoman's Whip	11. Route 66	21. Batman	31. Balloon Race
2. Rest for 25 Minutes	12. Rest for 5 Minutes	22. Rest for 5 Minutes	32. Route 66
3. 1909 Illions Carousel	13. Commotion Ocean	23. Bizzaro	33. Kontiki
4. Hurricane Falls	14. Rest for 5 Minutes	24. Rest for 35 Minutes	34. Stampede Bumper Cars
5. Rest for 10 Minutes	15. Balloon Race	25. Stampede Bumper Cars	35. 1909 Illions Carousel
6. Crime Wave	16. Rest for 25 Minutes	26. Typhoon	36. Batman
7. Rest for 45 Minutes	17. Cannonball Falls	27. Rest for 5 Minutes	37. Big Kahuna
8. Kontiki	18. Rest for 20 Minutes	28. Blizzard River	38. Go Home
9. Blizzard River	19. Big Kahuna	29. Hurricane Falls	
10. Rest for 15 Minutes	20. Rest for 10 Minutes	30. Wait 5 Minutes	

The results presented indicate a significant amount of wait-time between rides. This is due to the amount of nausea generated from the rides before the wait time and the user has to relax for a certain amount of time to recover.

There is a large difference in the number of generations needed to find an optimal solution between speed-pass and non speed-pass trips. This is because the speed-pass users are scheduled for many more rides than the non speed-pass user. This results in the algorithm having to run longer to find an optimal solution.

There is a significant difference between the number of generations needed to find an optimal solution between the fictitious park and the actual park. The actual park has several times the number of rides as the fictitious park and therefore needs several times the number of generations to reach an optimal solution.

The ability to create a optimized schedules for amusement parks relies on information which the park maintains and input parameters from the user about their preferences. Wait times for a ride and how nauseous the ride will make the user is normally not released to the public by the parks for marketing purposes. The authors used estimates for these values, based on past experiences, to approximate an actual representation of an amusement park.
There are many more ways to rank the enjoyment of an attraction through user input. The user could choose to input more advance information such as their favorite rides and which rides they want to avoid. Weights could be applied to these new values to allow the fitness function to take these new preferences into account.

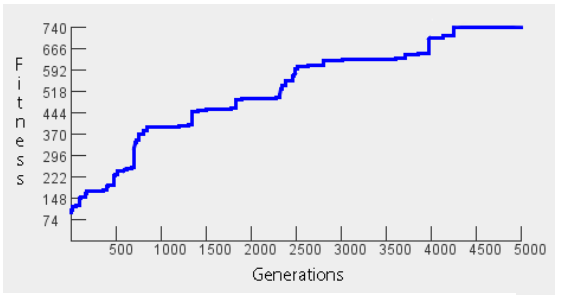

Fig. 2 Six Flags non speed-pass: Fitness vs Generations

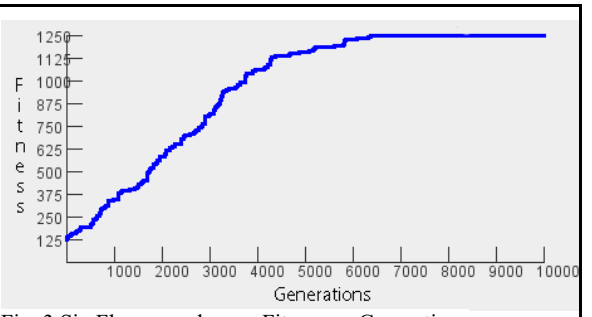

Fig. 3 Six Flags speed-pass: Fitness vs Generations

Speed-pass: Yes; Generations:10000; Nausea(1-10): 3 - High nausea tolerance; Ride Diminishing Returns(1-10):8 - Enjoys duplication of rides in the schedule; Fitness of Best Chromosome:1250.3451862175998

Decoded Chromosome to Actual Rides:

1. Blizzard River 2. Route 66 3. Rest 5 Minutes 4. Blizzard River 5. Rest 5 Minutes 6. Balloon Race 7. Rest 20 Minutes 8. Mind Scrambler 9. Rest 15 Minutes 10. Batman 11. Rest 15 Minutes 12. Monsoon Lagoon 13. Rest 20 Minutes 14. Big Kahuna	15. Rest 5 Minutes 16. 1909 Illions Carousel 17. Rest 20 Minutes 18. New England Skyway 19. Kontiki 20. Rest 10 Minutes 21. Stampede Bumper Cars 22. Route 66 23. Rest 10 Minutes 24. Splashwater Falls 25. Rest 20 Minutes 26. Geronimo Falls 27. Hurricane Falls 28. Rest 5 Minutes	29. Swiss Family Toboggan 30. Rest 20 Minutes 31. Commotion Ocean 32. Rest 20 Minutes 33. Big Kahuna 34. Rest 10 Minutes 35. Crime Wave 36. Rest 10 Minutes 37. Splashwater Falls 38. Rest 20 Minutes 39. Hurricane Falls 40. Rest 5 Minutes 41. Catwoman's Whip	42. Rest 35 Minutes 43. Cannonball Falls 44. Rest 15 Minutes 45. Cyclone 46. Rest 20 Minutes 47. Bonzai Pipelines 48. Rest 15 Minutes 49. Hurricane Falls 50. Kontiki 51. Balloon Race 52. Balloon Race 53. Go Home

4. Conclusions

In today's day and age many companies use smartphone applications to solicit advertisements to the user. This is facilitated through massive amounts of data collected from the user. Businesses know what you like, what you are likely to buy, where you are and who you are. Here we are giving the user the power to create their own suggestions. Prior to arrival, a visitor can set their preferences and when they arrive, load up a application on their phone and allow it to create an optimized schedule based on those preferences. At every ride it should allow for a ranking of the ride and how it made the user feel. The information collected should then be shared with the park's network to keep an average of statistics for each ride. The park could then keep track of wait times for rides throughout the day to help optimize the application. This will allow for the parks management to gauge which rides have the longest lines and help visitors schedule the most enjoyable trip possible while improving patron distribution throughout the park and reducing wait times for all visitors

References

1. Stein, O., How to solve a semi-infinite optimization problem, European Journal of Operational Research, Vol. 223(2), December 2012, pp.312–320
2. Martello S., Toth P., Knapsack problems: Algorithms and Computer Implementation, Wiley, New york,1990
3. Garey M. R., Johnson D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman & Co., San Francisco, 1979
4. Olsen, Anne L, Penalty Functions and The Knapsack Problem, Proceedings of the First IEEE Conference on Evolutionary Computation 1994, pp.554-558
5. Cheng, W., Shi, H., Yin, X., Li, D. ,An Elitism Strategy Based Genetic Algorithm for Streaming Pattern Discovery in Wireless Sensor Networks, IEEE Communications Letters, April 2011, Vol.15(4), pp.419-421
6. Lipowski, A., Lipowski, D., Roulette-Wheel Selection via Stochastic Acceptance, Physica A-statistical Mechanics and Its Applications, 2012, Vol. 391(6), pp.2193-2196
7. Thierens, D., Adaptive Mutation Rate Control Schemes in Genetic Algorithms, Proceedings of the 2002 Congress on Evolutionary Computation, May 2002, Vol.1, pp.980-985
8. Micheli, G.D., Synthesis and Optimization of Digital Circuits, McGraw-Hill, New York 1994
9. Forrest, S., Genetic Algorithms, ACM Computing Surveys, Vol. 28, No. 1, March 1996, pp.77-80
10. Forrest, S., Genetic Algorithms: Principles of Natural Selection Applied to Computation, Science, August 1993, Vol. 261, pp.872-878
11. Juang, C.-F., Temporal problems solved by dynamic fuzzy network based on genetic algorithm with variable-length chromosomes, Fuzzy Sets and Systems, Vol. 142(2), March 2004, pp.199-219
12. De Falco, I., Della Cioppa, A., Tarantino, E., Mutation-based genetic algorithm: performance evaluation, Applied Soft Computing Journal, May 2002, Vol.1(4), pp.285-299
13. Brie, A., Morignot, P., Genetic Planning Using Variable Length Chromosomes, In ICAPS 2005, pp.320-329